Solving vehicle routing problem with tabu search algorithm

Thian Shan You
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malayisa
tp051932@mail.apu.edu.my

Beh Chi Hao
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malayisa
tp064850@mail.apu.edu.my

Lau Shan Hui
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malayisa
tp060254@mail.apu.edu.my

Zailan Arabee bin Abdul Salam
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malayisa
zailan@apu.edu.my

Lau Wai Cong
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malayisa
tp060063@mail.apu.edu.my

***Abstract*—Tabu Search (TS) algorithm is one of the popular metaheuristics applied in Vehicle Routing Problem (VRP). In this paper, we look at an implementation of VRP using the TS algorithm and modify its parameters to observe the results of the changes. Similar works to this project is reviewed, the implementation of TS and VRP used was discussed and modifications to the source code was made. Three tests conducted looked at the max iteration, Tabu tenure, and expanding the problem parameters, and the results showed that optimal parameters could be obtained for smaller problem sizes, and as the problem size increases, the effectiveness of the TS algorithm decreases.**

Keywords—Tabu search algorithm, vehicle routing problem, tabu list, tabu tenure

I. Introduction

Vehicle Routing Problem (VRP) is an integer programming problem purposed to fulfill demands of customers by a fleet of vehicles. The objective of VRP is to find the most optimal routes or least cost vehicle route for multiple vehicles to visit a set of locations to fulfill the demands of customers. Some constraints could be added such as capacity constraints and time windows. Since VRP is a generalization of TSP, therefore VRP will be reduced into TSP whenever there is only one vehicle. VRP plays an important role in the sectors of distribution, logistics and transportation such as food distribution, courier service and routing of public transportation [1][2].

Tabu Search (TS) is one of the popular metaheuristics implemented for optimization problems. The basic concept of TS is to ban the moves that were performed before and moves without improvement will not be taken as consideration in the action to avoid being stuck in local minimums. A tabu list is short-term memory to store a certain number of moves made, preventing the algorithm from revisiting previous solutions. Meanwhile, tabu tenure (tt) stands for the number of iterations that a move remains in the tabu list as those moves listed in the tabu list cannot be visited. By applying TS, local optimums are avoided through the selection of non-improving solutions. Tabu list is implemented to enhance the performance by preventing the cycles and looping back to the previous solutions. In addition, TS is free to be implemented in both discrete and continuous solutions [3].

II. Literature Review

This section looks at the related works done on VRP using TS algorithm over the years.

Wassan [4] used an advanced TS implementation called Reactive Tabu Search (RTS) to solve VRP. Although TS is believed to perform well with excellent diversification and intensification strategies, it was found that it may worsen the process if there are too many diversifications or intensifications. The performance and effectiveness of TS is highly relative to the tt value. RTS included two mechanisms to solve this issue, the first mechanism generated an automated tt that remained unchanged throughout the search process while the second mechanism would remove the search process from its current location if repetitions of solutions was detected. This mechanism is called Hashing Function Search (HFS. Hence, it can be concluded that a moderate tt value is necessary to carry out a smooth search process.

Arockia et al [5] used a TS algorithm to compare its effectiveness with Simulated Annealing (SA) when solving Capacitated VRP (CVRP). The study added an additional component of different initial strategies to the TS algorithm until a superior initial strategy was found. The comparison between TS and SA algorithms found that TS was more suited to smaller-sized problems while SA was better at solving medium-sized problems.

Cui-hua Guan et al [6] utilized a TS algorithm in VRP to to enhance the viable starting solution. 2-opt algorithm is implemented to generate neighborhood solutions followed by aspiration standard to update the solution. The tabu property of the first element in the tabu list would be removed in accordance with the FIFO queue concept whenever all candidate items are tabu. This procedure will only terminate unless the desired outcome is obtained. TS is proofed to resolve VRP successfully and achieving worldwide availability.

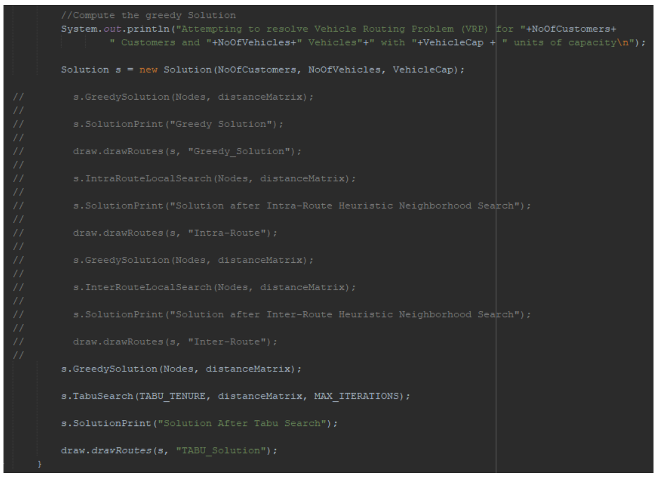
III. Materials and Methods

A. *Algorithm Implementation*

The implementation of VRP using TS algorithm was found on GitHub and was developed by the author 'nimich'. The implementation was built on Java 8 using IntelliJ and the code is run in a single class called VRP.java. The implementation first calculates a greedy solution, and three other heuristic algorithms were tested against it. The heuristic algorithms are Intra-route Search, Inter-route Search and Tabu Search. The code then prints the distance cost from each strategy in the developer console and creates 4 images to show the results of the resolution [7]. However, this paper is focusing on Tabu Search only, so a few modifications were made to the algorithm.

B. *Algorithm Modifications*

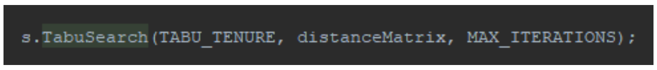
To conduct the algorithm modifications, the project was imported to the Apache Netbeans 12.3 IDE.



```
        //Compute the greedy Solution
        System.out.println("Attempting to resolve Vehicle Routing Problem (VRP) for "+NoOfCustomers+
                " Customers and "+NoOfVehicles+" Vehicles"+" with "+VehicleCap + " units of capacity\n");

        Solution s = new Solution(NoOfCustomers, NoOfVehicles, VehicleCap);

//          s.GreedySolution(Nodes, distanceMatrix);
//
//          s.SolutionPrint("Greedy Solution");
//
//          draw.drawRoutes(s, "Greedy_Solution");
//
//          s.IntraRouteLocalSearch(Nodes, distanceMatrix);
//
//          s.SolutionPrint("Solution after Intra-Route Heuristic Neighborhood Search");
//
//          draw.drawRoutes(s, "Intra-Route");
//
//          s.GreedySolution(Nodes, distanceMatrix);
//
//          s.InterRouteLocalSearch(Nodes, distanceMatrix);
//
//          s.SolutionPrint("Solution after Inter-Route Heuristic Neighborhood Search");
//
//          draw.drawRoutes(s, "Inter-Route");
//
        s.GreedySolution(Nodes, distanceMatrix);

        s.TabuSearch(TABU_TENURE, distanceMatrix, MAX_ITERATIONS);

        s.SolutionPrint("Solution After Tabu Search");

        draw.drawRoutes(s, "TABU_Solution");
    }
```

Fig. 1. Excluded code for greedy, intra-route and inter-route search

In the VRP.java class, the code related to the greedy solution, intra-route search and inter-route search between were commented off to have the program show the results of the Tabu Search optimizations only.



```
s.TabuSearch(TABU_TENURE, distanceMatrix, MAX_ITERATIONS);
```

Fig. 2. Modified arguments for TabuSearch method

Other modifications made to the code was the name change of *TABU_TENURE* from *TABU_Horizon* for clarity and the addition of an integer argument called *MAX_ITERATIONS* to the *TabuSearch* Java method in the VRP.java class. The purpose of this change is to move the max iterations parameter out of the *TabuSearch* method so that it can be placed closer to the other parameters which are going to be modified.

C. *Implementation Parameters*

The parameters for the implementation were categorized into the problem parameters, which is concerned with the VRP parameters, and the TS parameters itself.

1) *Problem Parameters:*

The problem parameters modify the scale of the VRP.

a) NoOfCustomers:

Sets the number of customers in the problem space, essentially the number of locations the vehicles will deliver to.

b) NoOfVehicles:

Sets the number of vehicles available for use. However, the algorithm may use less vehicles than the specified parameter.

c) VehicleCap:

Sets the carrying capacity of each vehicle.

2) *TS Parameters:*

There are only two parameters for the TS algorithm found in this implementation.

a) TABU_TENURE:

Determines the number of iterations a move should stay banned in the iteration. The parameter should be limited to less than or equal to the value of 20 to avoid runtime errors.

b) MAX_ITERATIONS:

The number of iterations the algorithm should run. The parameter should be limited to 100,000 as this implementation will simplify the VRP to a Travelling Salesman Problem in which only 1 vehicle is used to move all locations in a sequential manner.

D. *Parameter Modification*

With the parameters of the implementation established, multiple tests were conducted by modifying the values of the parameters. Four tests of the implementation were done on a Windows PC with an 8-core processor and 16 gigabytes of RAM to ensure consistency between the results. Each test involves modifying one or more parameters and running the implementation a total of 10 times to get an average of the distance cost. 10 runs of the implementation on default parameter settings were conducted as a control sample for comparison. With the control results, a percentage difference between the default parameters and the new parameters was obtained. A detailed description of the 3 tests conducted will be discussed in the next section.

IV. Results and Discussions

A. *Increasing Max Iterations*

TABLE I. Difference of Distance Cost When Max Iteration, (i) = 200 and 10000

No	Distance Cost		Difference of cost	
	i = 200	*i = 10000*	*Value*	*Percentage (%)*
1	647	606	41	6.8
2	659	565	94	16.6
3	652	590	62	10.5
4	640	507	133	26.2
5	658	629	29	4.6
6	652	509	143	28.1
7	650	622	28	4.5
8	652	569	83	14.6
9	658	517	141	27.3
10	625	616	9	1.5
Average	649.3	573	76.3	13.3

i: Max Iterations

The initial max iteration, i is 200 and the average distance cost is 649.3. As displayed in table above, the distance cost decreases when max iteration increases. When max iteration is increased to 10,000, an improvement of 13.3% was found in the average distance cost. Hence, it can be concluded that a higher max iteration results to a lower cost. However, the limitation of this is that this implementation is unable to accept a value of more than 100,000.

B. *Modifying Tabu Tenure Value*

The tests conducted here involves modifying the tt value by increasing it or decreasing it. The max iteration for the runs here uses the modified value from the previous test.

TABLE II. DIFFERENCE OF DISTANCE COST WHEN TABU TENURE (TT) = 10 AND 15

No	**Distance Cost**		**Difference of cost**	
	tt = 10	***tt = 15***	***Value***	***Percentage (%)***
1	606	509	97	19.1
2	565	570	-5	-0.9
3	590	548	42	7.7
4	507	532	-25	-4.7
5	629	523	106	20.3
6	509	514	-5	-1.0
7	622	513	109	21.2
8	569	586	-17	-2.9
9	517	536	-19	-3.5
10	616	538	78	14.5
Average	573	536.9	36.1	6.7

tt: Tabu Tenure

In Table II, a comparison of the tt = 10 and tt = 15 is made. The average distance cost when tt = 10 is 573 while the average distance cost for tt = 15 is 536.9. The percentage difference between the two averages are 6.7%. Conclusions can be drawn from results is a slight increase in tt can shows minor improvement to cost.

TABLE III. DIFFERENCE OF DISTANCE COST WHEN TABU TENURE (TT) = 10 AND 5

No	**Distance Cost**		**Difference of cost**	
	tt = 10	***tt = 5***	***Value***	***Percentage (%)***
1	606	615	-9	-1.5
2	565	647	-82	-12.7
3	590	627	-37	-5.9
4	507	629	-122	-19.4
5	629	635	-6	-0.9
6	509	592	-83	-14.0
7	622	629	-7	-1.1
8	569	630	-61	-9.7
9	517	615	-98	-15.9
10	616	600	16	2.7
Average	573	621.9	-48.9	-7.9

tt: Tabu Tenure

In Table III, the comparison here is between tt = 10 and tt = 5. The average distance cost for the default value is 573 while the average is 621.9 when tt = 5. The average distance cost became 7.9% worse after the change in tt value. Hence, we can conclude that the slight decrease in tt value negatively impacts the performance of TS.

From the changes made, we can conclude that the optimal parameter for the TS algorithm at 30 customers and 10 vehicles is tt = 15 and max iterations = 10,000. The difference in average would be from 649.3 down to 536.9, which is a 20.9% improvement in distance cost.

C. *Expanding the Problem Parameters*

After testing the changes in value for the max iterations and tt, the problem parameters were expanded to see if the modified parameters will yield the same improvements. Below are the changes made to the problem parameters:

- NoOfCustomers: 30 to 120
- NoOfVehicles: 10 to 140

TABLE IV. DIFFERENCE OF DISTANCE COST BASED ON DEFAULT AND MODIFIED PARAMETERS

No	**Distance Cost**	
	tt = 10, i = 200 (Default)	***tt = 15, i = 10000 (Modified)***
1	2171	2170
2	2172	2171
3	2172	2170
4	2170	2170
5	2171	2171
6	2170	2171
7	2170	2170
8	2170	2170
9	2170	2171
10	2170	2170
Average	2170.6	2170.4

tt: Tabu Tenure, i: Max Iterations

In Table IV, the comparison made is between tt = 10, iteration = 200 and tt = 15, iteration = 10,000. The average distance cost for both are quite similar with the default parameters at 2170.6 and modified parameters at 2170.4. Hence, we can conclude that Tabu Search is more effective when given a small problem size and a decrease in effectiveness can be noted as the problem size increases.

V. CONCLUSIONS

In conclusion, we can learn that from the modifications of these parameters, the TS algorithm requires high iteration counts to yield optimal performance and the Tabu tenure needs to be tweaked depending on the problem to determine the optimal value. However, these modifications only apply if the problem size is small enough. It was found that as the problem size increases, the effectiveness of the algorithm decreases.

The limitation of this study is the simplified implementation of VRP and TS algorithm as there were only a few parameters that could be manipulated, and the values of the parameters could only go to a certain limit before a runtime error, or the results devolved the VRP into a Travelling Salesman Problem. The VRP used here was also a classic version of the problem and new variants of the problem were developed to meet various changing real-world factors.

To improve the quality of this study, additional parameters to VRP and the TS algorithm could be added to address those limitations, or an implementation of the problem and algorithm could be developed from the ground up.

VI. REFERENCES

[1] P. Manzano. “What Is The Vehicle Routing Problem (VRP)?” Routific. https://blog.routific.com/what-is-the-vehicle-routing-problem [accessed Sep. 15, 2021]

[2] “Vehicle Routing Problem.” Google OR-Tools https://developers.google.com/optimization/routing/vrp [accessed Sep. 15, 2021]

[3] F. Liang. “Optimization Techniques — Tabu Search.” Towards Data Science. https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25 [accessed Sep. 16, 2021]

[4] N. Wassan. “A Reactive Tabu Search for the Vehicle Routing Problem.” *Journal of the Operational Research Society*. [accessed Sep. 16, 2021]

[5] A. Arockia, M. Lochbrunner, T. Hanne and R. Dornberger. “Benchmarking Tabu Search and Simulated Annealing for the Capacitated Vehicle Routing Problem.” *The 4th International Conference on Computers in Management and Business.* [accessed Sep. 16, 2021]

[6] C. Guan, Y. Cao and J. Shi. "Tabu Search Algorithm for Solving the Vehicle Routing Problem." *Third International Symposium on Information Processing.* [accessed Sep. 16, 2021]

[7] “Vehicle Routing Problem.” Github https://github.com/nimich/VehicleRouting [accessed Sep. 17, 2021]